Tecnológico de Monterrey

Actividad: Implementación de una Linked List

Leon Daniel Vilchis Arceo A01641413

Álvaro González Martínez A01646343

Gregorio Alejandro Orozco Torres A01641967

Daniel Hernández Gutiérrez A01640706

Lorenzo Orrante Román A01641580

16 de septiembre del 2025

Programación de Estructuras de Datos y Algoritmos Fundamentales

Grupo 604

Linked List:

Es una estructura de datos lineal, que está formada por nodos, cada nodo tiene dos partes: El dato que es el que almacena el valor que queremos guardar y la referencia que es la dirección del siguiente nodo en la lista. Además la lista se compone de head, tail y length, head apunta al primer nodo, tail apunta al último y length nos indica cuántos nodos hay. Si la lista está vacía head = nullptr. Es importante mencionar que a diferencia de los arrays, la memoria de los nodos no está en posiciones contiguas, sino que cada nodo se conecta con el siguiente.

Sus métodos principales son:

Insertion: que es agregar un nodo al inicio, al final o en alguna posición específica.

Deletion: Eliminar algún nodo, dado un valor o posición

Traversal: Permite visitar cada nodo en orden desde inicio hasta final

Search: Permite checar si el nodo que buscamos existe.

VS Array:

A diferencia de un array, la linked list no requiere saber el número de elementos que va a llevar, se pueden ir agregando o eliminando nodos sin necesidad de redimensionarla.

Además, en un array insertar o eliminar algún elemento implica mover muchos valores en la memoria. En una linked list, solamente vas modificando las referencias entre los nodos, lo que hace que estas operaciones sean más rápidas en posiciones intermedias.

También algo muy importante es que en una linked list vas creando los nodos según los vas necesitando, mientras que en un array tienes que reservar un espacio fijo grande desde el inicio, aunque no termines usándolo.

Otro punto clave es que la linked list aprovecha mejor la memoria, sin embargo, los arrays siguen siendo más convenientes cuando se necesita acceso directo y rápido por índice, ya que en una linked list hay que recorrer nodo por nodo hasta llegar al valor.

Complejidad de insert, delete y search:

insertAtBeginning: Tiene una complejidad de O(1), ya que no recorre toda la lista y solo actualiza los punteros del head. El tiempo de ejecución es constante independiente del tamaño.

insertAtEnd: Este método cuenta con una complejidad de O(1), tail hace que acceda por el final y así evitar recorrer toda la lista, así también logrando que tenga un tiempo de ejecución constante.

insertAtPosition: Este método debe recorrer la lista hasta la posición que se busca en el peor escenario recorre toda la lista, es por ello que su complejidad es O(n). El while (currPosition < pos) puede correr hasta n cantidad de veces.

deleteValue: Es un método de complejidad O(n), ya que busca el valor que se debe de eliminar recorriendo la lista y en el peor de los casos llegara al final de ella. El while (curr.next) puede ejecutarse hasta n cantidad de veces.

search: Este método es de complejidad O(n), esto es porque recorre toda la lista hasta poder encontrar el valor especificado y haciendo que potencialmente pueda recorrer todos los nodos. El while (curr) puede ser ejecutado hasta n cantidad de veces.

Link del video:

https://youtu.be/2k5qG2Kw1JY?si=PvN6mViUQW3afLhA